Grid Troubleshooting Issues

Brian L. Tierney, Dan Gunter, Jason Lee

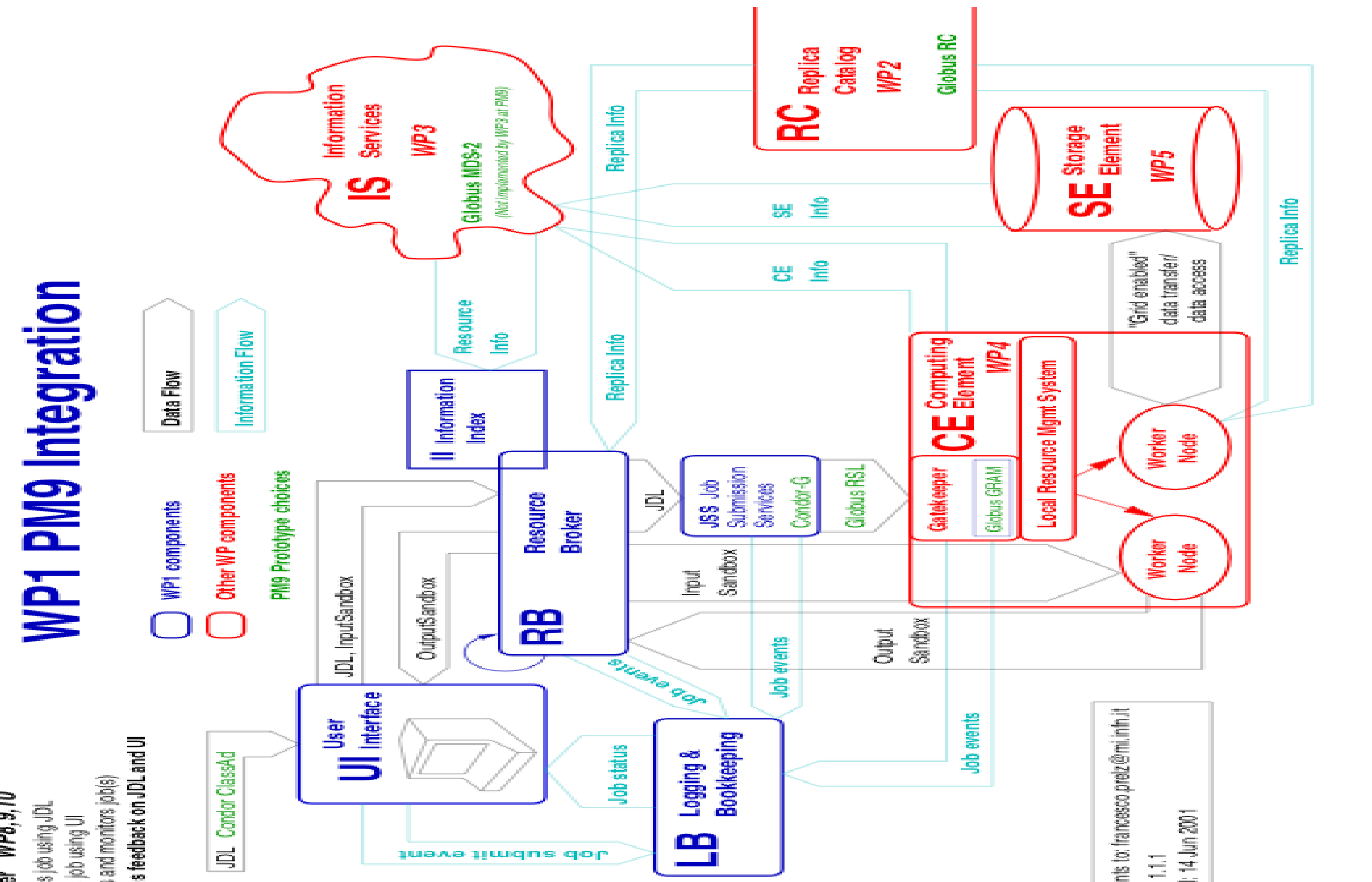
bltierney@lbl.gov, dkgunter@lbl.gov
http://dsd.lbl.gov/DMF/

Distributed Systems Department
Lawrence Berkeley National Laboratory

The Problem

- Assume a Grid job is:
  - submitted to a resource broker, uses a reliable file transfer service to copy several files, then runs the job.
- This normally takes 15 minutes to complete. But…
  - two hours have passed and the job has not yet completed
- What, if anything, is wrong?
  - Is the job still running or did one of the software components crash?
  - Is the network particularly congested?
  - Is the CPU particularly loaded?
  - Is there a disk problem?
  - Was a software library containing a bug installed somewhere?

Example: EU DataGrid Components



Good Troubleshooting is Essential

- Grids are getting larger and more complex
  - More components = Higher probability of failure

- Individual components may be very robust
  - But, Combination of many robust components not necessarily robust
  - Failures can be very hard to detect (eg.: TCP problems)

- Complex troubleshooting is part of the fundamental nature of a service oriented architecture

The Need for Better Troubleshooting

- From the Grid3 Lessons Learned document
  - http://www.ivdgl.org/grid3/documents/document_server/uploaded_documents/doc--760--Lessons_V8.doc
- **30% failure rate** for ATLAS and CMS simulations
  - especially for the long jobs (more than 4 or 5 hours)
  - 90% of the failures were caused by problems at the computing site:
    - disk filling errors, gatekeeper overloading or network interruptions
- “another reason for the **job failures came the middleware itself**; many glitches were detected when we ran large numbers of production jobs…”
- “There were many instances of very heavy CPU load on site head/gatekeeper nodes. This was in some cases attributable to the number of jobs being submitted to a site. In other cases it appears to be due to the monitoring services running on the gatekeeper node. In other cases it looks like some daemons may have “run amok”. At present ***our diagnostic tools are lacking*** for being alerted to and being able to understand the causes.”
- “While we were successful in running all applications fairly stably across many of the Grid3 sites in most cases ***each application had to be debugged on each site*** to achieve this.”
- From the Grid3 Summary Presentation:
  - “**The trouble shooting capabilities of the user need to be improved**
    - Currently users have log files to help, but it’s **time consuming**.
    - Improve the tools to diagnose problems”

Why is troubleshooting hard?

- There are frameworks available to access system monitoring information
  - Ganglia, MonaLisa, MDS, etc.
- There are several methods to handle application instrumentation
  - syslog, log4j, SvPablo, printf, etc.
- But, no integration between the two:
  - No common data models
  - No common data formats
  - No common aggregation and collection mechanisms
  - No common analysis and visualization tools

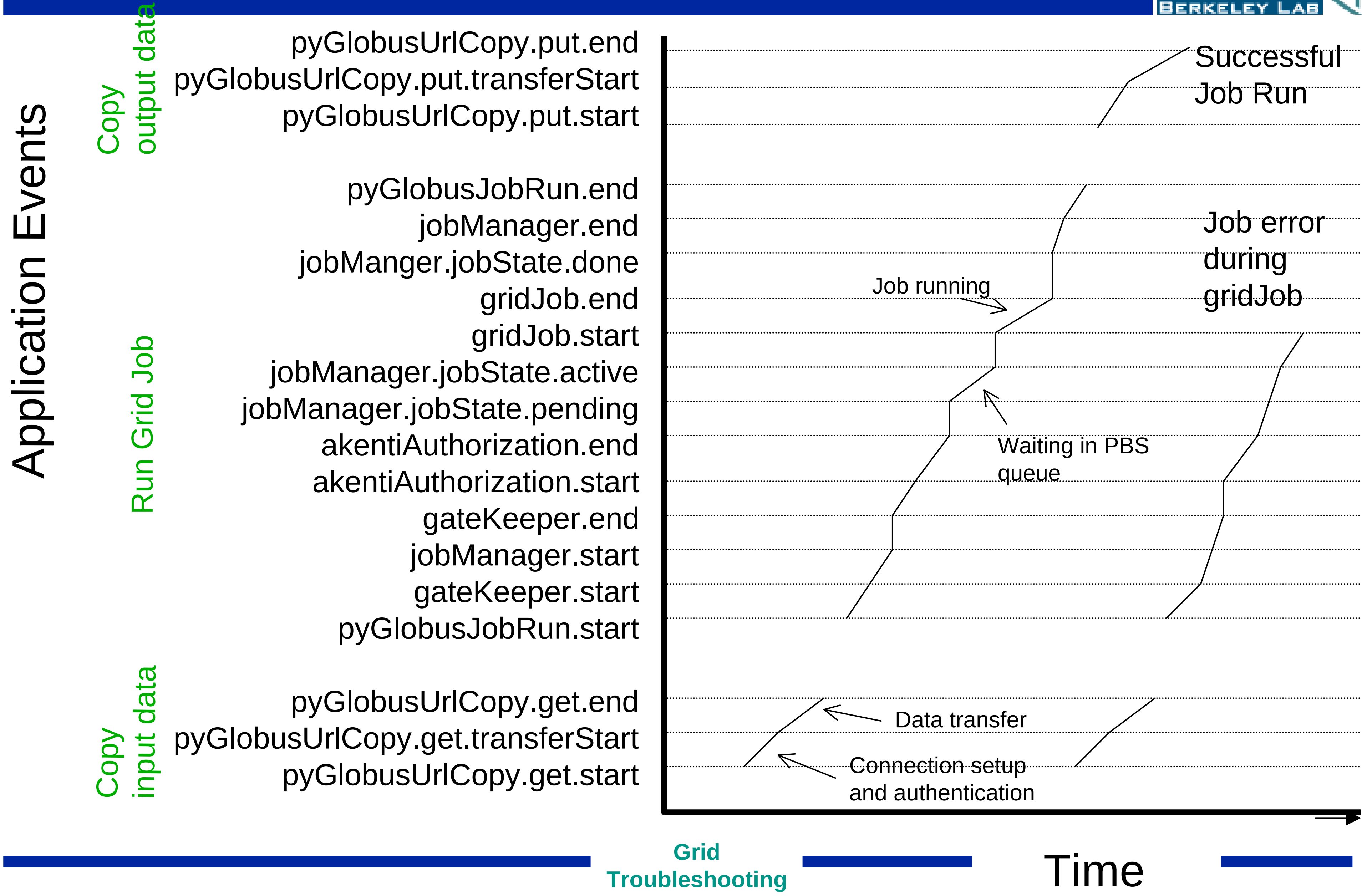
Solution

- An **End-to-End** instrumentation AND monitoring framework:
  - instrumentation tools (application, middleware, and OS monitoring)
  - host and network sensors (host and network monitoring)
  - sensor management tools
  - monitoring data publication service
  - monitoring data archive service
  - analysis and visualization tools
    - Ability to correlate data from many sources
  - protocols for describing, exchanging and locating monitoring data
- This is the goal of the LBNL NetLogger Toolkit

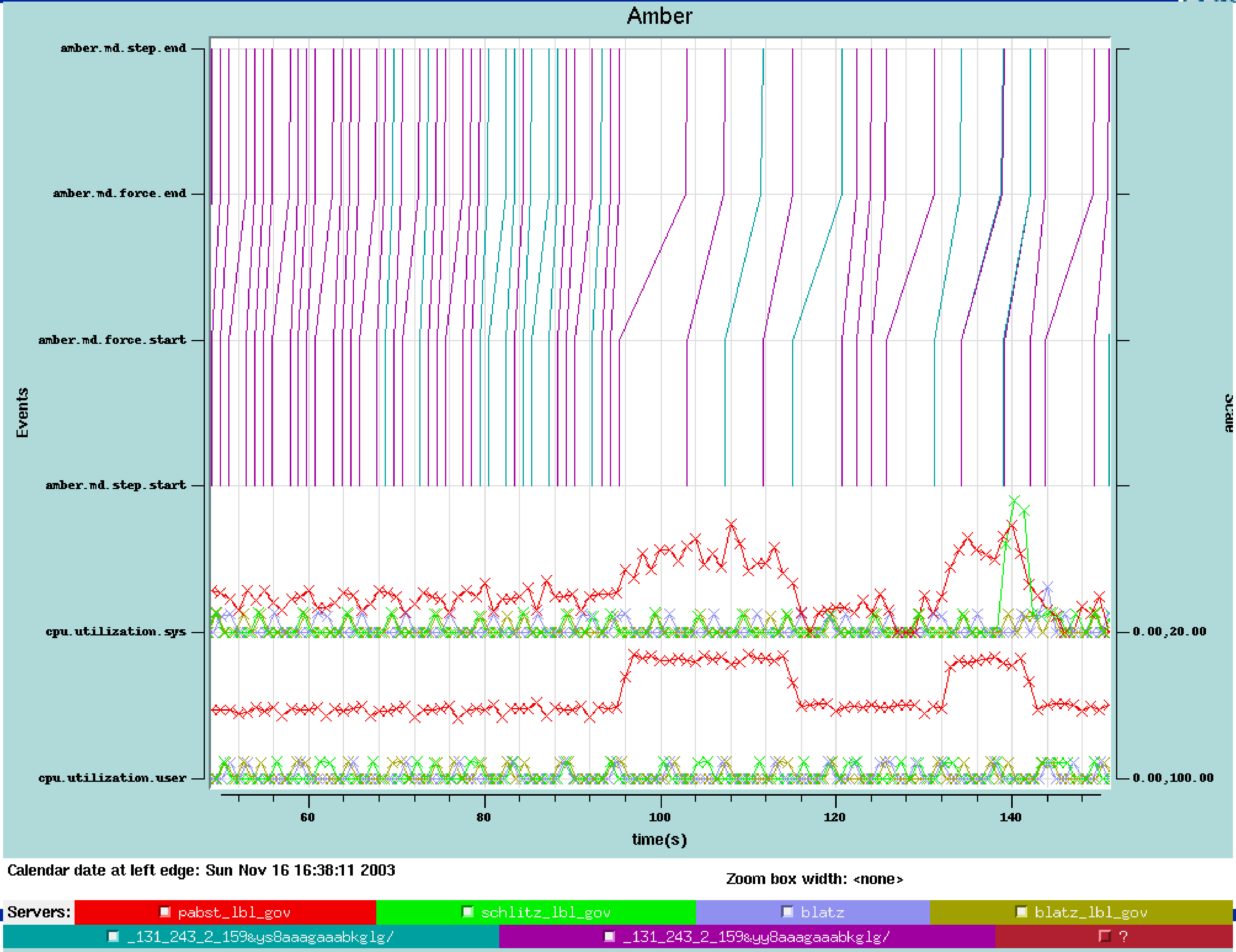
Several Missing Pieces

- Grid Workflow ID’s
  - Needed to correlate events
- Common data model
- Automatic instrumentation
- Data discovery
  - need to easily locate all the instrumentation and monitoring data related to given Grid Job.
- Better Analysis Tools

Using Grid Workflow IDs to Generate a Job "Lifeline"



Correlation of Application Instrumentation and CPU Monitoring



Calendar date at left edge: Sun Nov 16 16:38:11 2003

Common Data Model

- A common log format very useful, but a simple common data model is a *fundamental requirement* for analysis of independent data sources.
  - Mapping between disparate data models is difficult
    - much more difficult than format translations
  - Needed to be able to perform relational DB queries
  - Needed for analysis tools

- Various GGF working groups are addressing this:
  - Discovery and Monitoring Event Descriptions WG
    - http://dsd.lbl.gov/damed/
  - Network Measurements WG
    - http://dsd.lbl.gov/NMWG/

Automatic Instrumentation

- Busy programmers rarely get around to properly instrumenting their code
- No standard instrumentation formats / methods
- We need automatic instrumentation tools that can be applied to deployed software components
  - Compiled languages (C, C++, Fortran, etc.)
    - tools should work on object files and not require access to source code if possible.
  - Interpreted languages (Python, Java, etc.)
    - introspection capabilities of the language can be used to provide run-time control of instrumentation points.
  - At a minimum, we need to wrap Grid components with simple start/stop instrumentation wrappers
  - Use of automatic methods helps enforce a standard data model and standard format
  - Use of visual programming languages / toolkits with built in instrumentation will help

What should be done in the short term?

- It is essential to be able to do Grid troubleshooting soon
  - Many application domains are now looking into using the Grid
  - 70% success rate is not good enough!
- Simple, high payoff items
  - Start using Grid Workflow IDs for correlating data
  - Wrap all Grid components with start/end instrumentation wrappers
- Less simple, but important items
  - Monitoring frameworks and instrumentation mechanisms should be merged into a common framework (e.g.: NetLogger)
  - Design a common data model for monitoring and instrumentation data
  - Use of higher level application frameworks will help get standard instrumentation deployed